The Human-Robot Interaction Operating System

Terrence Fong
NASA Ames Research Center
Moffett Field, CA 94035 USA
terry.fong@nasa.gov

Clayton Kunz
NASA Ames Research Center
Moffett Field, CA 94035 USA
ckunz@mail.arc.nasa.gov

Laura M. Hiatt
Carnegie Mellon University
Pittsburgh, PA 15213 USA
lahiatt@cs.cmu.edu

Magda Bugajska
Naval Research Laboratory
Washington, DC 20375 USA
magda.bugajska@nrl.navy.mil

ABSTRACT
In order for humans and robots to work effectively together, they need to be able to converse about abilities, goals and achievements. Thus, we are developing an interaction infrastructure called the "Human-Robot Interaction Operating System" (HRI/OS). The HRI/OS provides a structured software framework for building human-robot teams, supports a variety of user interfaces, enables humans and robots to engage in task-oriented dialogue, and facilitates integration of robots through an extensible API.

Categories and Subject Descriptors
I.2.9 [**Robotics**]: Autonomous vehicles; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

Keywords
human-robot interaction, interaction infrastructure, robot architecture, multi-agent system

1. INTRODUCTION
There are many forms of human-robot teams, each with their own benefits and drawbacks[20]. The most common form is teleoperation, in which a robot is used as a "tool". Although teleoperation is appropriate for extreme and unknown environments, it generally binds system capabilities to the operator's skill and performs poorly when communication is delayed or bandwidth-limited.

At the opposite end of the control spectrum are "autonomous" robot systems, in which the robot operates independently to achieve high-level goals and the human functions as a monitor or supervisor. Autonomous systems can multiply effort (e.g., a single user commanding multiple robots) and can function when the communication between human and robot is poor, but are often brittle and incapable of handling unforeseen events.

Regardless of form, human-robot teams only perform well when humans and robots are able to work in a synergistic manner. In particular, system configurations that enable humans and robots to communicate (conversing about goals, abilities, plans and achievements) and to collaborate (jointly solving problems) are less brittle, more robust, and better performing than those that do not.

1.1 Peer-to-peer human-robot interaction
In our work, we are investigating how *peer-to-peer* human-robot interaction (HRI) can facilitate communication and collaboration[10]. Our approach is to develop an interaction infrastructure that enables humans and robots to communicate and work as partners, in a manner inspired by human work crews. In our system, for example, robots are able to ask task-oriented questions of the human in order to obtain assistance when they are in trouble.

A key feature of our approach is that humans and robots coordinate their actions through dialogue. This helps contextual and situational awareness to be maintained across the team. Dialogue also enables humans and robots to support one another. This allows better application of the different strengths and capabilities of humans and robots, and helps balance workload.

1.2 A novel infrastructure for HRI
In the following sections, we present a novel interaction infrastructure, the "Human-Robot Interaction Operating System" (HRI/OS), which is inspired by the *collaborative control* model[11]. The HRI/OS is an agent-based system that provides a structured framework and set of interaction services for human-robot teams. We have designed the HRI/OS to facilitate the integration of a wide range of user interfaces and robots through an extensible API.

We begin by describing the types of tasks and the teamwork model that the HRI/OS supports. We then discuss the design of the HRI/OS: its agent-based structure and the primary agents that comprise the system. Finally, we examine the HRI/OS in practice, focusing on a use case involving multiple humans and robots.

Human Robot Interaction '06 Salt Lake City, UT USA

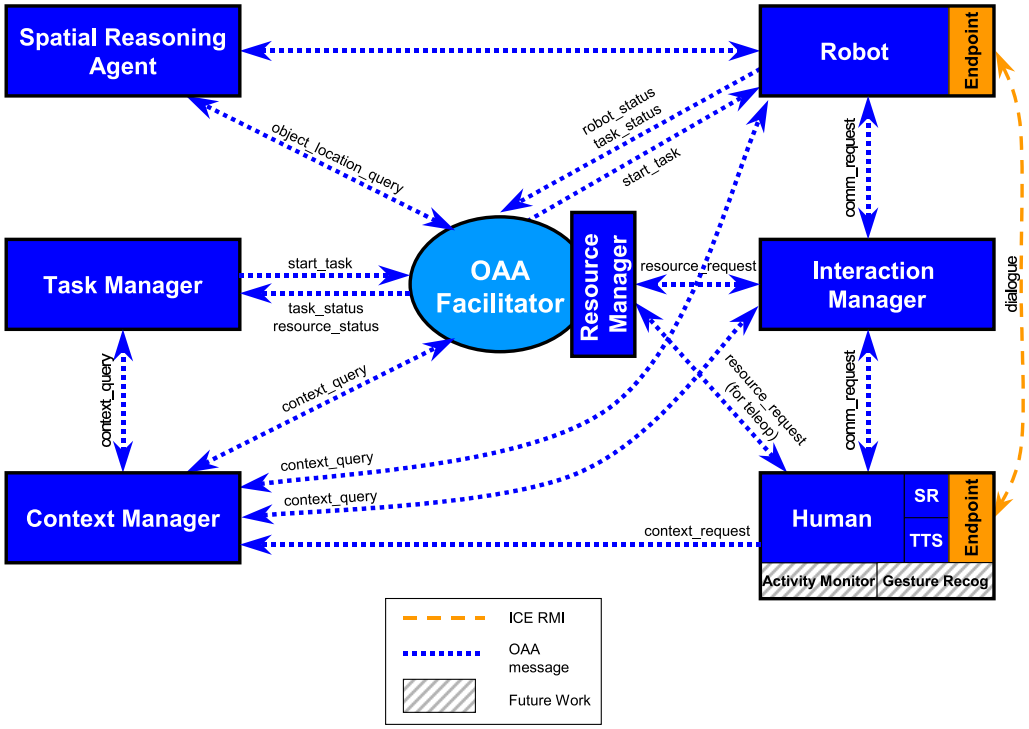

Figure 1: The Human-Robot Interaction Operating System (HRI/OS) is an agent-based system.

2. APPROACH

2.1 Operational Tasks

The HRI/OS is designed to support the performance of *operational tasks*, which are tasks that are concrete, well-defined, narrow in scope and amenable to joint human-robot performance. In space exploration, for example, operational tasks include: shelter and work hangar construction, piping assembly and inspection, pressure vessel construction, habitat inspection, and *in-situ* resource collection and transport[9].

A pragmatic approach to performing operational tasks is to specify a high-level set of operations for humans and robots to execute in parallel and then use interaction to provide detail and to resolve problems that arise during execution. This approach is similar to how human teams function, particularly construction and maintenance work crews.

2.2 Teamwork Model

The teamwork model we use assumes that each member has a set of skills and resources that they contribute to the team. In the HRI/OS, high-level tasks are delegated by a centralized executive to embodied agents (human or robot), which it believes capable of satisfying the task and who are not engaged in other work. Agents execute these tasks, performing detailed planning and monitoring as necessary.

During execution, if an agent finds that its skills or resources prove to be inadequate to the task, it tries to resolve the situation through dialogue (i.e., rather than immediately reporting task failure). For example, a robot that has difficulty interpreting camera data might ask a human to lend his visual processing ability to the task. This often allows tasks to be completed in spite of limitations of autonomy.

If a human requests assistance from a robot, the robot suspends its task before responding. After providing assistance, the robot then resumes the task. With our teamwork model, a robot can only be interrupted by one human at a time. That is, a human cannot interrupt a robot while it is already providing assistance to another human. Instead, the human must wait until the robot becomes available.

3. DESIGN

3.1 Agent-based architecture

The HRI/OS is an agent-based system that incorporates embodied agents (humans and robots) and software agents (Figure 1). Embodied agents operate in the physical world and describe their skills at a coarse level, rather than with the detail typically used by robot planners. Software agents do not directly change the environment and are designed to provide limited services. For example, an agent that tracks objects may provide pose information without handling coordinate frame transformations, which could be provided by other supporting agents. This design approach helps improve flexibility while reducing system brittleness.

The current implementation uses the Open Agent Architecture (OAA)[6] for inter-agent communication and delegation. Agents communicate via OAA messages, which are delegated via a centralized facilitator. Direct, point-to-point communication (used primarily to transport binary data) is performed using the "ICE" middleware[13].

In general, when an agent makes a request to the HRI/OS, it does not know which agent (or agents) will satisfy the request. Such anonymity reinforces the concept of peer-to-peer HRI, since any agent making a request to the system

must necessarily treat humans and robots in the same manner. Delegation is performed by OAA, assisted by a domain-specific resource manager (Section 3.3). In the current implmentation, for example, the resource manager considers spatial location to improve delegation of physical tasks.

In order for HRI to be productive, humans and robots need to be able to communicate efficiently and effectively. Consequently, the HRI/OS provides cognitive models and spatial reasoning capabilities that allow humans to use natural, spatial language (e.g., "move the light to the left of the box"). Additionally, when a robot asks for help, it is important that its request be given to a human with appropriate expertise and ability to respond. Thus, the HRI/OS incorporates an interaction manager (Section 3.4), which takes into consideration the human's situation (workload, location, available user interfaces, etc.) before involving him in dialogue.

The HRI/OS is similar in some respects to *interaction infrastructures* that support non-traditional human-computer interaction[18, 22, 27]. In particular, the HRI/OS provides a variety services commonly associated with infrastructures, such as data and event distribution to heteogeneous clients. The HRI/OS, however, differs from infrastructures because it uses a task delegation model and because the "devices" (humans and robots) are embodied.

A number of HRI architectures have recently been proposed for human-robot teams[4, 7, 16, 23]. The HRI/OS, however, differs from these architectures in three significant ways. First, the HRI/OS is explicitly designed to support human-robot collaboration across multiple spatial ranges and team configuration. Second, the HRI/OS assumes that humans and robots will work on tasks in parallel, with only loose coordination between them. Finally, the HRI/OS allows robot control authority to pass between different users (i.e., no operator has exclusive "ownership" of a robot) to improve flexibility and situational response.

3.2 Task Manager

The Task Manager (TM) is responsible for coordinating and managing the execution of operational tasks. It does this by decomposing the overall goal of the system into high-level tasks, which are assigned to humans or robots for execution. Only a single task is assigned to a given agent at a time. Unlike traditional executives, the TM does not know anything about low-level task details. Instead, it relies on each agent to work in a distributed, independent manner, managing and monitoring their own task execution.

The TM is implemented in the Task Description Language (TDL), a superset of C++ that allows for principled task execution, coordination and management[21]. TDL allows the tasks to be represented with appropriate inter-task constraints, such as serialization. For example, Figure 2 shows a construction task in which two panels must both be mounted before the seam between them can be welded and then inspected. Such constraints are important because agents may not be immediately available to perform tasks, or may need to suspend execution in order to assist another agent.

To assign a task, the Task Manager contacts the Resource Manager (Section 3.3) in order to find an agent capable of performing the work. The RM either immediately assigns the task to an agent, in which case the TM begins monitoring the status of the task, or notifies the TM that no agent is currently available. If this is the case, the TM waits until some agent is available, then again makes its request.

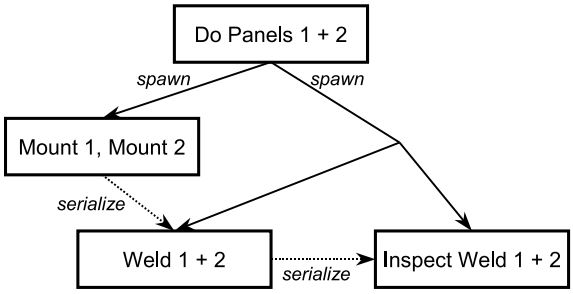

Figure 2: Decomposition of a welding task: panels 1 and 2 are mounted, the common seam is welded, and then inspected. Parts of this task can be performed in parallel, others require sequencing.

The TM is designed to recover in the face of error or task failure. If any task fails, the TM will create another instance to retry the task. The TM also provides functionality for reacting to feedback from a task via task monitoring. For example, if the result of a weld inspection indicates that the weld is inadequate, the TM will respawn another weld and inspect pair, repeating this process until the inspect task indicates that the weld has been successfully completed.

As currently implemented, the TM provides fairly simplistic task management. One improvement would be for the TM to work in conjunction with the Resource Manager in order to predict agent availability. This would especially be helpful when a human requests assistance and there are multiple robots capable of performing the work, but who are currently engaged on other tasks. Another improvement would be for the TM to reassign a task whenever an agent performing that task has to suspend execution (e.g., in order to respond to dialogue).

3.3 Resource Manager

The Resource Manager (RM) processes all agent requests, prioritizing the list of agents to be consulted when a task needs to performed or a question answered. Unlike facilitation in most agent systems, the RM performs delegation using multiple criteria that vary with time and situation, rather than simple capability match making. In particular, in addition to request/capability matching, the RM considers a variety of factors including availability, physical location, workload, past performance, etc.

The current RM is implemented as a collection of OAA meta-agents. Each meta-agent is designed with limited scope and reasons only about a single criteria. This approach allows addition of new criteria (i.e., as new meta-agents) without modification of existing code, and thus is more flexible and extensible than monolithic design. Specifically, domain and goal specific knowledge and selection criteria can be added (or removed) as needed.

Robot_A: "I need help inspecting a weld."
Robot Agent → Interaction Manager:
`comm_request (Robot_A, help, weld)`
Interaction Manager → Resource Manager:
`agent_request(help, weld)`
Resource Manager → Interaction Manager:
`User_1`
Interaction Manager → User_1:
`message_notification`
User_1 → Interaction Manager:
`message_request`
Interaction Manager → User_1:
`comm_request(Robot_A, help, weld)`
User_1 → Interaction Manager:
`comm_response(Robot_A, endpoint_address)`
Interaction Manager → Robot_A:
`comm_response(User_1, endpoint_address)`
Robot_A and User_1 begin dialogue

Figure 3: Message exchange resulting from "Robot_A" requesting help.

In the future, we intend to add predictive capabilities to the RM. One way to do this would be to employ a planner that can reason about resource allocation and usage. This would enable delegation to consider not only the current availability and situation, but also estimated performance (e.g., time required to service request).

3.4 Interaction Manager

The Interaction Manager (IM) coordinates dialogue-based interaction between agents. The IM gives each agent the ability to communicate with other agents: to ask for help, to provide help, etc. The HRI/OS currently supports graphical and speech user interfaces, but other modalities (e.g., visual gesturing) will also be included in the future.

With the HRI/OS, whenever an agent needs to communicate, it sends its request to the IM. The IM queries the Resource Manager for a list of agents capable of handling the request and then contacts the first one. If the agent is a robot, the IM immediately forwards the request. Otherwise, if the agent is a human, the IM notifies the human that there is a pending communication and waits for the human to respond. If the request is time-critical and the receiving agent fails to respond, the IM tries the next agent on its list.

For example, Figure 3 shows the message exchange between agents when a robot requests help. Once both parties of the conversation are engaged, the IM steps out and allows direct dialogue to begin. The IM only intervenes if the requesting agent is unable to satisfy its request via the dialogue. In this case, the IM repeats the dialogue process using the next agent from the Resource Manager's list.

In the HRI/OS, agents communicate with one another via point-to-point "dialogue endpoints", which are implemented using the "ICE" middleware[13]. Endpoints allow agents to send a variety of data (text, images, sensor readings, etc.) to each other. This approach provides more flexibility and better performance (quality of service, low-latency transfer, etc.) than OAA's centralized, text-based messaging.

3.5 Context Manager

In a complex agent system, keeping track of the activities and state of agents over time is a difficult task. This is particularly true when multiple agents operate in parallel and when activity is observed remotely (e.g., via user interfaces). Thus, to facilitate situational awareness, we have developed a Context Manager (CM). The CM keeps track of everything that occurs while the system is running: task status and execution, agent activities, agent dialogue, etc. Then, when agents have need to recall history, they can query the CM for a summary of information.

The CM continuously tracks system state by storing events and data generated by agents in a time-stamped archive. In many respects, this approach is similiar to distributed data logging mechanisms, such as described in [25]. When an agent requests information concerning another agent, a task, or a particular system event, the CM searches the archive and identifies which data are potentially relevant. It then processes this data to create a summary response.

At present, the CM performs summarization by: (1) pruning duplicate or similar archive data (e.g., a dialogue event is reported by both the sender and the receiver); (2) sorting messages by time; and (3) making use of dialogue summaries provided by agents. Future work on the CM will make this process more sophisticated by extracting dialogue summaries directly from the archive, summarizing the execution of tasks over time, etc.

The CM is related in many ways to several other event tracking systems. Martin et al. (2003) developed an interaction and collaboration system that tracks agent location, activity, role, etc. and is capable of summarizing/identifying event patterns[15]. Similar to this is the Meeting Browser, which "eavesdrops" on all dialogue that occurs during a meeting and summarizes it for later use[26]. Other research has focused on automatic definition, identification, and summarization of evolving events[1].

3.6 Robot Agent

Robot Agents (RA's) provide an interface between robot controllers and the HRI/OS. RA's process requests received from other agents, manage task execution, and engage in dialogue with other agents. The RA is extensible: the current C++ API contains both a common core (robot-independent) and custom (robot-specific) methods.

During nominal operation of the HRI/OS, high-level tasks are assigned to robots by the Task Manager. Upon receiving an assignment, the RA decomposes the high-level task into a sequence of primitives to be executed. Once the robot starts executing the sequence of primitives, it can only be interrupted, or engage in dialogue, at a breakpoint. A breakpoint is defined as a point in execution where: (1) the task can be resumed without requiring preservation of context/state; (2) the task cannot not proceed due to failure or error; or (3) robot is not working (i.e., waiting for a task to be assigned).

A key feature of the RA is that it provides methods for asking questions of humans and for handling the responses received. Whenever a robot has a question to ask, it sends a message to the Interaction Manager. A message is defined

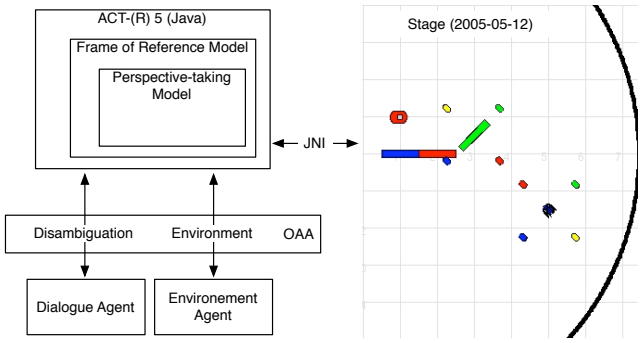

Figure 4: Spatial Reasoning Agent architecture

by query attributes (priority level, expiration time, etc.), query type (y/n, multiple choice, etc.) and message-specific data (image, text, etc).

The RA provides a variety of run-time support functions. On start-up, each RA registers the capabilities of its associated robot with the Resource Manager. During operation, RA's periodically broadcast event messages (robot state, task progress, etc.) to other agents. Finally, the RA is responsible for coordinating control authority switching between "system operation" (Task Manager driven) and "user mode" (human has direct authority of the robot).

3.7 Spatial Reasoning Agent

When human-robot teams perform operational tasks (construction, maintenance, etc.) understanding and communicating spatial dialogue plays a significant role[17]. In particular, when humans and robots operate in a shared workspace, robots must be able to understand how humans perceives space and the relative positions of objects around them. To give robots this ability, we are developing computational cognitive models for spatial perspective-taking and frames of reference[5, 10, 14, 24].

The spatial reasoning agent (SRA) is used to resolve spatial ambiguities in human-robot dialogue. The current implementation of the SRA resolves frame of reference ambiguities including *ego-*, *addresse-*, *object-*, and *exo-centric* references. We use the Java version of the ACT-R[3] cognitive architecture system, jACT-R, and the Player/Stage environment [12] to model and resolve frames of reference and perspective-taking (Figure 4).

Whenever human-robot dialogue involves spatial language, the HRI/OS forwards a spatial reasoning request to SRA as a parameter set (speaker, addressee, type of command, reference objects, and frame of reference). For example, the command "Robonaut, move to the left of Box 1" from an astronaut to the robot "Robonaut", is passed to the SRA as (astronaut, Robonaut, move, Box 1, left, ego). The SRA then transforms the spatial dialogue into a geometric reference using the cognitive model.

To resolve spatial ambiguities, the cognitive model is used to perform a "mental simulation" of the interaction. First, the model executes productions, which obtain information about the relevant objects (current pose of the speaker, addressee, etc.) and stores them as chunks in declarative memory. This information is than used to model the real-world in the Stage simulator (Figure 5a).

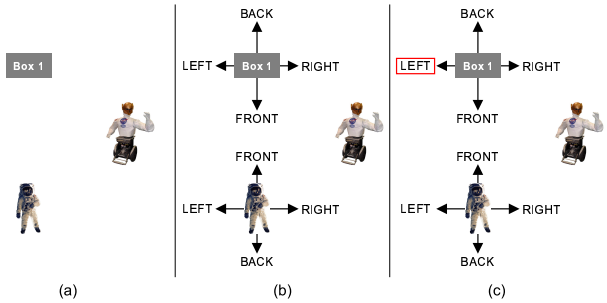

Figure 5: Disambiguation process: (a) Configuration of mental simulation in Stage, (b) Assignment of frames of reference relevant objects, (c) Resolution of ambiguous location.

When a mental simulation is created, a frame of reference is assigned to the reference object. The model uses three frames: *ego*, *referent*, and *exo*. The ego frame of reference is the traditional egocentric frame that can be applied to agents and objects. The referent frame of reference is used when: (1) the object or agent referred to does not have its own frame of reference and (2) a frame which is a mirror of another agent's frame of reference has to be used. Finally, the exo frame of reference represents the exocentric (world) coordinate system.

Productions retrieve the information necessary to choose a frame and location to which it is to be assigned to, and place the chosen frame at the desired location. Continuing the example, the referent frame oriented towards the astronaut would be placed on Box 1 (Figure 5b).

Finally, the desired perspective is assigned to the world based on the perspective parameter of the request. In the example given above, the world would be perceived from the astronaut's location. At this point, the SRA is able to fully resolve the spatial language ("left of Box 1") in both local (relative to astronaut) and global coordinates (Figure 5c).

3.8 Human Proxy

In order for the HRI/OS to work with humans and robots in the same "manner", humans need to appear as software agents, just as robots do. To do this, we have developed a human proxy agent that represents user capabilities and accepts task assignments, much in the same way that the Robot Agent does for robots. Human proxies have been used in numerous HRI architectures, such as [7] and [16].

Human proxy agents publish task capabilities, domains of expertise in which they can be called upon to provide help via dialogue, and provide health monitoring feedback that can be used by other agents to track the overall progress of the task. Human proxy agents make use of user interfaces to communicate with the users that they represent, and make use of the Interaction Manager to manage peer-to-peer dialogue with other agents.

The relationship between human proxy agents and the users they represent is complicated by the fact that humans have more internal state than robots. A human proxy agent might accept a task on behalf of its user, only to have the human refuse to perform the task, or worse, agree to perform the task and then ignore it. Human proxy agents are implemented in Java and typically run on a wearable computer, which communicates via a wireless data network and which integrates with health monitoring sensors.

3.9 User Interfaces

When humans work with robots, they will have different interaction capabilities depending on a variety of factors: team configuration (e.g., shared space vs. remote), worksite environment, communication links, etc. For example, a suited astronaut can currently only communicate with others (humans and robots alike) using speech, while a human inside a spacecraft or habitat will likely have access to multiple computer displays and pointing devices. Thus, we have designed the HRI/OS to support a variety of user interfaces and languages (currently C++ and Java).

Speech services. The HRI/OS currently provides two facilities for speech interaction: text-to-speech (TTS) and small vocabulary speech recognition (SR). The TTS and SR agents are currently implemented using the Microsoft Speech SDK and run on Windows platforms. In addition, the SR agent uses the Ariadne Spoken Dialog System, which is a domain independent toolkit for speech-based applications[8].

Several instances of these agents might be active at any given point in time, depending on the number of humans active in a given work scenario. For example, if the scenario includes a single voice loop shared by all users, then there may be only be a single TTS agent (which synthesizes speech for everyone) and individual SR agents for each user.

Graphical user interfaces (GUI). GUI's provide traditional pointer and window driven interaction. GUI agents provide a broader range of dialogue support than speech interfaces, including images, video, and 2D/3D display. The HRI/OS currently provides several standard GUI's for system operation and simple peer-to-peer query/response dialogue.

The *Map* GUI shows the current spatial distribution of humans and robots. It allows its user to initiate dialogue with specific agents as well as to switch robot control authority (from Task Manager to user). The Map also displays status and event information received from each agent: task in progress, health data, etc.

Each robot typically also provides a *Teleop* GUI through which the human can directly effect control. Some robot GUI's support supervisory control; other GUI's support manual control. With the HRI/OS, Teleop GUI's can only be used after a request is made to the robot to switch to user control. This allows the robot to gracefully suspend any work that it is engaged in prior to relinquishing control.

4. CASE STUDY

4.1 Use Case: Seam Welding and Inspection

We are currently using the HRI/OS to study a use case that centers on seam welding by a team of multiple humans and robots. Seam welding is a basic task that will be required for building and maintaining structures on planetary surfaces[19]. For example, linear welds might be used to construct pressure vessels, work hangers, and emergency shelters too large to raise into space in one piece.

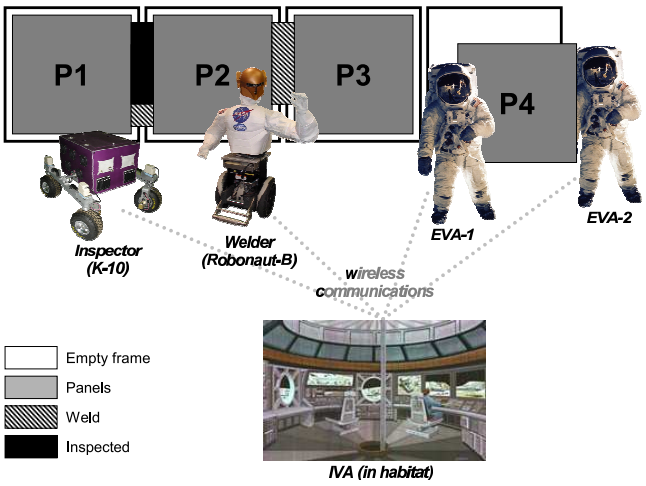

Figure 6: Seam welding with a human-robot team. Left to right, (a) an inspection robot (K-10) verifies the weld between panels P1 and P2, (b) a welding robot (Robonaut-B) welds panels P2 and P3, (c) two suited astronauts (EVA1 and EVA2) carry a panel to the frame for welding, (d) an astronaut (IVA) inside a habitat provides remote support.

In our study, humans work side-by-side and remotely with two robots, the K-10 rover (NASA Ames) and Robonaut-B (NASA JSC) to weld* panels to a truss structure (Figure 6). K-10 is a low-cost, highly maneuverable mobile robot designed for human-paced interaction (i.e., it is capable of human walking speeds) and is equipped with a camera and high-intensity spotlight. Robonaut-B is a two-armed humanoid robot mounted on a Segway RMP base and carries a welding tool[2].

In a typical work scenario, two suited astronauts (EVA1 and EVA2) act as *master welders* and provide initial panel mounts (e.g, tack welds). The robots perform two types of tasks. Robonaut-B works as a *junior welder* and welds seams between mounted panels. K-10 serves as a *seam inspector* and inspects the quality of the welds. A third astronaut (IVA), who is located inside a habitat, interacts with and supports the remote workcrew (both humans and robots) via wireless communications. Throughout the task, humans and robots work in parallel, supporting each other as necessary.

4.2 HRI/OS Execution

The use case provides numerous opportunities for dynamic and flexible human-robot interaction. For example, a variety of communication acts are useful: human generated commands, questions from the robots to the human, etc.

*Because it is not our goal to improve robotic welding, a "mock welding" process (e.g., spray painting) is being used for weld seaming during the study.

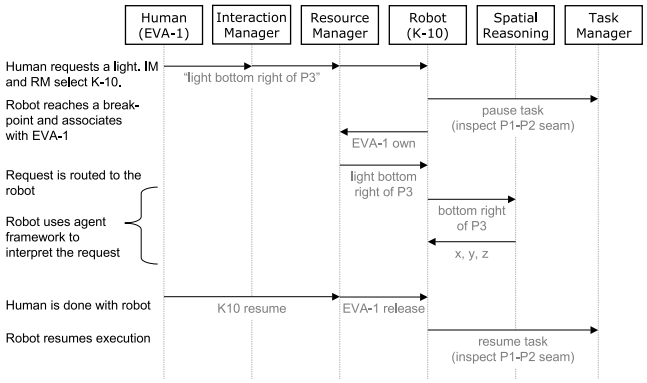

Figure 7: Execution sequence of robot supporting a human.

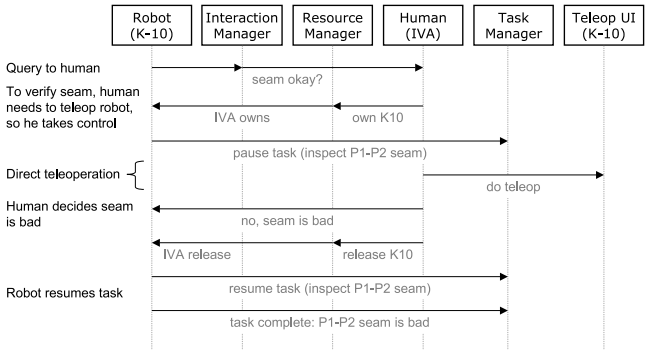

Figure 8: Execution sequence of human supporting a robot.

Additionally, humans and robots may interact in a shared space (i.e., the workcrew), or remotely (e.g., IVA may teleoperate or provide assistance to a robot).

4.2.1 Robot supports human

Consider the situation shown in Figure 6. The two suited astronauts are placing a panel (P4) onto the frame when a problem occurs: EVA-1 has trouble aligning the panel with the previous panel (P3). To remedy this problem, EVA-1 uses the HRI/OS to request that a light be pointed at the bottom right corner of P3. Figure 7 shows the resulting execution sequence.

The human's request is initially passed to the IM and RM, which decides that the K-10 robot is best able to handle the request. When K-10 reaches a breakpoint in its work, it suspends its current task and associates itself with EVA-1. K-10 then uses the underlying agent framework to handle the request (i.e., to interpret the spatial language). Because K-10 and EVA-1 are associated, K-10 continues to support the human (handle requests) until released. At that point, K-10 resumes execution of its previous task.

A key point about this approach is that the human-robot team, EVA-1 and K-10, is formed dynamically and lasts only as long as needed. This approach allows the HRI/OS to provide flexible response to unforseen problems. This approach contrasts strongly with other systems, in which a human typically has long-term "ownership" of a robot (i.e., robot is a tool). Furthermore, because the robot(s) that support a human are assigned (by the RM) based on availability, capability, and a variety of other constraints (e.g., spatial location), the HRI/OS ensures that resources are used effectively in response to time-varying situational needs.

4.2.2 Human supports robot

A distinguishing feature of the HRI/OS is that it allows humans to support robots. For example, if K-10 is inspecting a seam and determines that it cannot adequately assess the quality, it can ask "is this seam okay" and send supporting inspection data (e.g., an image of the seam in question). Figure 8 shows the resulting execution sequence.

K-10's request is initially passed to the IM and RM, which decides that IVA (the human in the habitat) is best able to handle the request. After reviewing the data sent by K-10, IVA decides that he needs additional information before answering the question. Consequently, he requests ownership of K-10 (which suspends its current task when it reaches a breakpoint), then teleoperates the robot (e.g., moving K-10's camera to obtain supplemental views of the seam). When IVA decides that the seam is bad, he answers the question and releases K-10, which then resumes execution.

There are several important points to note. First, when a robot asks for help, the human that is asked to respond is chosen based on numerous factors including expertise, availability, etc. This approach is dual to the way robots are selected to support humans. Second, because users often have different interface capabilities, the form of the information passed may vary. For example, a suited astronaut with a speech interface will receive spoken dialogue, whereas a human with multiple graphical displays will receive images, charts, etc. Finally, the HRI/OS enables switching of control between "system" and "user" modes. This facilitates human-robot teaming in a manner similar to human work crews, i.e., when a junior crew member (the robot) has trouble, and expert (the human) can step in and take over.

5. CONCLUSION

The HRI/OS is a novel infrastructure for HRI that supports peer-to-peer dialogue and provides a variety of services, including task delegation, resource management, and human proxy. The HRI/OS enables humans and robots to work as partners, supporting one another as they perform operational tasks. Moreover, by incorporating computational cognitive models, the HRI/OS helps make human and robot more understandable to each other, so that interaction becomes more natural.

Our long-term goal is to extend the HRI/OS to support large human-robot teams. To do this, the Task Manager will need to be able to dynamically reason about resource usage and availability. Additionally, although the current system tracks robot progress, a similar facility is needed for tracking human work. This will require more extensive cognitive modeling and the use of activity monitoring techniques. Finally, in order for team members to communicate effectively and understand one another, especially when help is being requested, mechanisms for establishing common grounding are needed.

6. ACKNOWLEDGMENTS

We would like to thank Dan Christian and Illah Nourbakhsh for contributing to the HRI/OS design. We would also like to thank Lorenzo Flückiger and Robert Burridge for integrating the K-10 and Robonaut-B robots into the HRI/OS. We would also like to thank David Lees, Pavithra Rajagopalan, Mikey Siegel, and Vinh To for developing the speech and graphical user interfaces. This work was sponsored by NASA grant HRT-ICP-04-0000-0155.

7. REFERENCES

[1] S. Afantenos, K. Liontou, et al. An introduction to the summarization of evolving events: Linear and non-linear evolution. In *International Workshop on Natural Language Understanding and Cognitive Science*, 2005.

[2] R. Ambrose, H. Aldridge, et al. Robonaut: Nasa's space humanoid. *IEEE Intelligent Systems Journal*, Aug 2000.

[3] J. Anderson and C. Lebiere. *Atomic components of thought*. Erlbaum, Mahwah, NJ, 1988.

[4] J. Bradshaw et al. *Software Agents*, chapter KAoS: Toward an industrial-strength open agent architecture. MIT Press, 1997.

[5] N. Cassimatis, J. Trafton, et al. Integrating cognition, perception, and action through mental simulation in robots. *Robotics and Autonomous Systems*, 49(1-2), Nov 2004.

[6] A. Cheyer and D. Martin. The open agent architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1):143–148, March 2001.

[7] W. Clancey et al. Automating capcom using mobile agents and robotic assistants. In *Proc. AIAA 1st Space Exploration Conference*, 2005.

[8] M. Denecke. Rapid prototyping for spoken dialogue systems. In *Proc. 19th International Conference on Computational linguistics*, 2002.

[9] T. Fong and I. Nourbakhsh. Interaction challenges in human-robot space exploration. *ACM Interactions*, 12(2):42–45, 2005.

[10] T. Fong, I. Nourbakhsh, et al. The peer-to-peer human-robot interaction project. In *Space 2005*, number AIAA 2005-6750. AIAA, 2005.

[11] T. Fong, C. Thorpe, and C. Baur. Collaboration, dialogue, and human-robot interaction. In *Proc. 10th International Symposium on Robotics Research*. Springer, 2001.

[12] B. Gerkey, R. Vaughan, and A. Howard. Player/Stage project: Tools for multi-robot and distributed sensor systems. In *Proc. International Conference on Advanced Robotics*. 2003.

[13] M. Henning. A new approach to object-oriented middleware. *IEEE Internet Computing*, 8(1), 2004.

[14] L. Hiatt, J. Trafton, et al. A cognitive model for spatial perspective taking. In *Proc. 6th International Conference on Cognitive Modelling*. 2004.

[15] C. Martin, D. Schreckenghost, et al. Aiding collaboration among humans and complex software agents. In *Spring Symposium*. AAAI, 2003.

[16] C. Martin, D. Schreckenghost, et al. An environment for distributed collaboration among humans and software agents. In *Proc. International Conference on Autonomous Agents and Multi-Agent Systems*, 2003.

[17] J. Reitsema, W. Chun, et al. Team-centered virtual interactive presence for adjustable autonomy. In *Space 2005*, number AIAA 2005-6606. AIAA, 2005.

[18] M. Roman, C. Hess, et al. Gaia: a middleware infrastructure to enable active spaces. *IEEE Pervasive Computing*, 1(4), 2002.

[19] C. Russell et al. Considerations of metal joining processes for space fabrication, construction and repair. In *Proc. SAMPE Technical Conference*. 1991.

[20] J. Scholtz. Human-robot interactions: Creating synergistic cyber forces. In A. Schultz and L. Parker, editors, *Multi-robot systems: from swarms to intelligent automata*. Kluwer, 2002.

[21] R. Simmons and D. Apfelbaum. A task description language for robot control. In *Proc. Conference on Intelligent Robots and Systems*, 1998.

[22] J. Sousa and D. Garlan. Aura: an architectural framework for user mobility in ubiquitous computing environments. In *Proc. IEEE/IFIP Conference on Software Architecture*, 2002.

[23] A. Tews, M. Mataric, and G. Sukhatme. A scalable approach to human-robot interaction. In *Proc. IEEE International Conference on Robotics and Automation*, 2003.

[24] J. Trafton, N. Cassimatis, et al. Enabling effective human-robot interaction using perspective-taking in robots. *IEEE Trans. on Systems, Man and Cybernetics, Part A*, 49(4), July 2005.

[25] H. Utz, G. Mayer, and G. Kraetzschmar. Middleware logging facilities for experimentation and evaluation in robotics. In *German Conference on AI*, 2004.

[26] A. Waibel, M. Bett, and M. Finke. Meeting browser: Tracking and summarising meetings. In *Broadcast News Workshop*. DARPA, 2003.

[27] T. Winograd. *HCI in the New Millennium*, chapter Interaction spaces for 21st century computing. Addison Wesley, 2001.